Universidade do Minho
Escola de Engenharia
Departamento de Informática

SIMPÓSIO DOUTORAL 2006

1 Identification

1.1 Name

Nuno F. Rodrigues, nfr@di.uminho.pt

1.2 Title

Generic Software Slicing Applied to Architectural Analysis of Legacy Systems

1.3 Supervisor

Prof. Luís Soares Barbosa

1.4 Starting Date

1, April, 2005

1.5 Ending Date

31, March, 2008

2 Abstract

2.1 Investigation Area

Formal Methods, Software Architectures.

2.2 Abstract

Program slicing techniques are widely used for identifying and isolating the parts of a program that depend on or are depended upon by a specific element (e.g. a variable, a routine). This PhD project aims to scale up such techniques, well-known at the micro, program-oriented level, to the macro, software architectural level. This research effort requires the development of generic slicing algorithms, applicable to large, multi-language, heterogeneous systems as well as new formal techniques to extract, represent and reasoning upon architectural information from legacy systems. The techniques and methods developed will be tested with real legacy data supplied by a leading software company.

2.3 Strategic Objectives

In a situation in which the only quality certificate of the running software artifact still is life-cycle endurance, customers and software producers are little prepared to modify or improve running code. However, faced with so risky a dependence on legacy software, managers are more and more prepared to spend resources to increase confidence on - i.e. the level of understanding of - their code. Actually, if forward software engineering can today be regarded as a lost opportunity for formal methods (with notable exceptions in areas such as safety-critical and dependable computing), reverse engineering looks more and more a promising area for their application. This is due to the complexity of reverse engineering problems and exponential costs involved.

In this context, the present PhD proposal has the following objectives:

- Development of formal, rigorous models to represent architectural information, with the precise identification, in legacy software, of components, services, behaviours and data clusters needed to create higher level abstractions of the target systems and guide their reconstruction.

- Development of wrappers and suitable extraction algorithms to retrieve architectural information from actual systems.

- Development of generic slicing techniques, and their mathematical derivation, to scale up slicing algorithms already used at the micro, program-oriented level, to the macro, software architectural level, in a language independent way.

- Application of such techniques to real case studies of large legacy software systems, i.e. systems that have been programmed in multiple languages, frameworks and database systems and consists of many thousands of lines of code.

- Development of prototypes of tools to assist the working software architect in the application of the techniques developed.

The starting point of this PhD project is an extensive, functional library of generic slicing algorithms, which covers (almost) all existing slicing approaches, which is currently being developed by the candidate as part of his MSc thesis. This research is being carried on in the context of the PURe Project (Program Understanding and re-engeneering: Calculi and Applications) initiated in October 2003, under FCT contract POSI/ICHS/44304/2002, coordinated by Luís Soares Barbosa.

3 Contributions

3.1 Major Scientific Contributions

HaSlicer. The HaSlicer is a tool that peformes both Haskell programs graphical visualization and diverse slicing operations. The tool is available online at http://labdotnet.di.uminho.pt/HaSlicing/HaSlicing.aspx.

Component Identification Through Program Slicing. Based on the the deliverables of the HaSlicer tool, in particular the intermediate representation of programs as instances a Functional Dependence Graphs (FDG), the student developed a method for software entities cluster discovery which leads to the identification of potencial functional components in legacy code.

3.2 Publications

Rodrigues, N. and Barbosa, L., Slicing Functional Programs by Calculation. In Proceedings of the Dagstuhl Seminar on Beyond Program Slicing, (Schloss Dagstuhl, Wadern, Germany, Nov. 6-11, 2005).

N. Rodrigues and L. S. Barbosa. Component identification through program slicing. In *Proc. Formal Aspects of Component Software (FACS'05)*. ENTCS (to appear), Elsevier, 2005.

N. F. Rodrigues, E. Sousa, J. Rodrigues, and F. Santos. A mobile systems interface protocol. In *Proc. Innovative Views of .NET Technologies (IVNET'2005)*.

N. Rodrigues and L. S. Barbosa. Prototyping concurrent systems in c omega. In *Proc. Innovative Views of .NET Technologies (IVNET'2005)*.

Nuno F. Rodrigues and Luís S. Barbosa. Architectural prototyping: From ccs to .net. *Electr. Notes Theor. Comput. Sci.*, 130:151-167, 2005.

4 Context

4.1 Scientific Context and Motivation

Program Slicing Program slicing is a technique for isolating the parts of a program that depend on or are depended upon by a specific element of the program. A program slice consists of portions of the original program that have some effect on a particular computed value, referred to as a *slicing criterion*. The effort of computing a program slices is called *program slicing*.

The first appearance of the term and the underlying discipline of *program slicing* is due to Weiser[11] in 1979. Weiser states that a slice corresponds to the mental abstractions that people make when they are debugging a program, and suggests the integration of program slicers in debugging environments.

Since the genesis of *program slicing*, several definitions of what a *program slice* is, or the best methods to obtain such units of this discipline have emerged from different areas in Computer Science. The major reason for this diversity is that particular program environments require different slicing approaches, reflecting different properties from the computed slices (see the extensive bibliography listing attached).

A relevant distinction in the process of obtaining program slices is that between *static slicing* and *dynamic slicing*. The former one relies on the objective observation of the source code, without making any assumptions about the input's of the program, whereas the latter computes a *program slice* regarding some input values of the analyzed program. Other relevant dual concepts in *program slicing* is *forward slicing* and *backward slicing*. the first starts from a given program point, and finds all program parts that depend on it, while the second starts from a given program point and finds all program parts upon which it depends. The intersection of a forward and backward slice, is called a program chop.

Originally, slicing techniques were developed to slicing within the scope of a single procedure in a procedural programming language, such as C. Later, generalizations were made to additionally allow for inter-procedural slicing, where several procedures are taken into account. Further generalizations include slicing of declarative and object-oriented languages. Various tools have been developed for slicing particular programming languages in the context of specific applications (e.g., CodeSurfer).

It was shown that program slicing can basically be expressed as a reachability analysis on so-called program dependence graphs. Thus, slicing can basically done in two phases: building a program dependence graph from program source code, and performing reachability analysis on the resulting graph. Slicing approaches differ in the particular dependence graphs being used, and in the programming paradigms for which they are suitable.

The bibliography on program slicing is quite vast; more then 250 fundamental references were evaluated by the candidate in a previous report. For a tutorial overview see e.g. [10, 5, 1]

Architectural Slicing for Systems' Reconstruction Software architecture[3, 9, 2] is becoming more and more an indispensable phase of design in the software development process as well as a fundamental analysis level in legacy systems understanding. The software architecture of a system captures, in a family of models, the high level, both static and dynamic (i.e., behavioural) structure of the main interacting components on which the systems relies. The diversity of architectural models proposed in the literature (e.g. [7, 12, 6]), ranging from semi-formal to formal, general purpose or domain oriented, witnesses the practical relevance of this area in the context of software engineering, but also the lack of consensual, widely used models (let alone formal foundations).

Applying slicing techniques to software architectural descriptions aims both to enhance system's understanding and analysis, as well as to make possible the extraction and reuse of particular architectural patterns. At this level, slicing can be made in terms of a (set of) component(s), connector(s), service(s) or behaviour(s) (e.g. as specified by *use cases*).

There exist very few works on the application of slicing techniques to the architectural level (but e.g. [8, 13]). Moreover, most of the less informal approaches to this problem discuss slicing on top of a representation model, failing to address the phases of information gathering and representation.

4.2 Detailed Objectives

Background Studies This first phase of the PhD project will be devoted to an extensive review of the literature in the following areas:

- Formal models for representation of architectural information. In particular, the expressive and calculational power of coalgebraic-based methods developed at Minho, in which the candidate has collaborated [4], should be analyse in detail and confronted with other alternatives. A few small case studies on gathering architectural information from real systems and representing it on the above mentioned models will be conducted.

- Slicing techniques. A substantial part of this is already done in the candidate's MSc work.

Architectural Slicing Techniques This is the core task in the whole PhD project. It encompasses the generalization of slicing concepts and methods to the macro, architectural level, the definition of a formal framework to define and reason about slicing techniques in a multi-lingual/multi-paradigm setting and the mathematical derivation of new slicing algorithms and their correctness proofs. During this task the basic architectural representation model should be defined. From the applicational point of view, a library of architectural slicing algorithms, formally documented, is the basic deliverable of this task. From a more foundational perspective, this work is expected to provide the basic elements of a generic theory of slicing.

Development of Architectural Extractors Once defined an architectural representation model suitable for slicing analysis, a number of target languages

and software platforms will be chosen so that suitable mechanisms for architectural information extraction, and corresponding prototype tools, could be defined in this task. This may include representatives of procedural languages (e.g. C++, VB or Java), database systems (e.g. SQL), document-oriented (e.g. XML), component frameworks (e.g. .Net) an scripting languages (e.g. Perl). The actual choice, however, depends on the actual case studies on architectural analysis supplied by the industry. This leads to the final phase of this project:

Architectural Analysis Case Studies Experimental research on the application of the techniques, algorithms and tools previously developed to the analysis and reconstruction of two real legacy systems. This assessment phase seems to be of fundamental importance to verify the suitability of the developed techniques and also to gather experimental data on the, still poorly understood, area of reverse engineering. It is expected that one of the case studeies will be supplied by a leading software company (already contacted); the other one will come from the open source community, in the context of an European project on certification of open source software.

4.3 Alternative Work

Addressing the particular problem of software architectural slicing (delivering real sub-architectures as a result), there are no approaches documented in the literature.

4.4 Main Bibliography

References

[1] In Mark Harman and Keith Gallagher, editors, *Information and Software Technology Special Issue on Program Slicing*, volume 40. Elsevier Science B. V., 1998.

[2] R. Allen and D. Garlan. A formal basis for architectural connection. *ACM TOSEM*, 6(3):213–249, 1997.

[3] Robert Allen and David Garlan. A formal basis for architectural connection. *ACM Transactions on Software Engineering and Methodology*, July 1997.

[4] L. S. Barbosa, M. Sun, B. K. Aichernig, and N. Rodrigues. On the semantics of componentware: a coalgebraic perspective. In Jifeng He and Zhiming Liu, editors, *Mathematical Frameworks for Component Software: Models for Analysis and Synthesis*, Series on Component-Based Development. World Scientific, 2004.

[5] David Wendell Binkley and Keith Brian Gallagher. Program slicing. In Marvin Zelkowitz, editor, *Advances in Computing, Volume 43*, pages 1–50. Academic Press, 1996.

[6] J. Fiadeiro and A. Lopes. Semantics of architectural connectors. In *Proc. of TAPSOFT'97*, pages 505–519. Springer Lect. Notes Comp. Sci. (1214), 1997.

[7] D. Garlan, R. Monroe, and D. Wile. ACME: An architecture description interchange language. In *CASCON'97*, 1997.

[8] Fumiaki Ohata, Kouya Hirose, Masato Fujii, and Katsuro Inoue. A slicing method for object-oriented programs using lightweight dynamic information. In *Proceedings of the 8th Asia-Pacific Software Engineering Conference*, December 2001.

[9] J.-G. Schneider and O. Nierstrasz. Components, scripts, glue. In L. Barroca, J. Hall, and P. Hall, editors, *Software Architectures - Advances and Applications*, pages 13–25, 1999.

[10] Frank Tip. A survey of program slicing techniques. Technical Report CS-R9438, Centrum voor Wiskunde en Informatica, Amsterdam, 1994.

[11] Mark Weiser. *Program slices: Formal, psychological, and practical investigations of an automatic program abstraction method*. PhD thesis, University of Michigan, Ann Arbor, MI, 1979.

[12] M. Wermelinger, A. Lopes, and J. Fiadeiro. A graph based architectural (re)configuration language. In V. Gruhn, editor, *Proc. of ESEC/FSE'01*. ACM Press, 2001.

[13] Jianjun Zhao. Applying slicing technique to software architectures. In *Proceedings of 4th IEEE International Conference on Engineering of Complex Computer Systems*, pages 87–98, August 1998.

5 Development

5.1 Work Macro-Planning

Schedule

The PhD works starting date is April, 2005, respecting the following scheduling :

T1 : Background Studies - 6 Months

T2 : Architectural Slicing Techniques - 12 Months

T3 : Development of Architectural Extractors and Architectural Analysis Case Studies - 12 Months

T4 : Final Thesis Writing - 6 Months

6 Evaluation

6.1 Accomplished Work Auto-Evaluation

As it would be expected during the first year of a PhD studies, the work developed so far is not yet focused towards the mainstream of the PhD objectives. Nevertheless, much of the studies and work developed over the foundations of program slicing will certainly by fruitful in the context of the overall Phd works.

6.2 Produced Documentation Auto-Evaluation

Much of the documentation produced so far reflects well the studies and work developed during this first PhD year. Due to the short period between publication and the present date, the impact of such publications can not yet be faithfully analysed.

7 Difficulties

There are no relevant difficulties to point out.